

REMARKS

In the Office Action, the Examiner rejected the claims under 35 USC §103. The claims have been amended to further clarify the subject matter regarded as the invention. The claim rejections are fully traversed below. New claims 53-59 have been added. Claims 1-59 are now pending.

Reconsideration of the application is respectfully requested based on the following remarks.

REJECTION OF CLAIMS UNDER 35 USC §103

In the Office Action, the Examiner rejected claims 1, 2, 7-20, 22-24, 26, 29, 35, 38, 43 and 49 under 35 USC §103 as being unpatentable over Preisler et al, U.S. Patent No. 5,675,803, ('Preisler' hereinafter) in view of Edwards et al, U.S. Patent No. 6,011,920, ('Edwards' hereinafter). Claims 18, 29, 38, and 49 are further rejected in view of Kirouac et al, U.S. Patent No. 5,155,847, ('Kirouac' hereinafter). Claims 3-6, 21, 25, 27, 28, 33, 34, 36, 37, 42, 44, 45, 47 and 48 are further rejected in view of Ahlin et al, U.S. Patent No. 5,321,840, ('Ahlin') hereinafter. Claims 31, 40 and 51 are further rejected in view of Glasser et al, U.S. Patent No. 5,793,980, ('Glasser' hereinafter). Claims 30, 32, 39, 41, 50 and 52 are further rejected in view of Halpern et al, U.S. Patent No. 6,282,711, ('Halpern' hereinafter). This rejection is fully traversed below.

System services such as those providing I/O functionality are often structured in the form of a stack. However, in the event of a system failure, one or more of the services in the stack may become inoperative. In fact, even one non-functional layer in the stack can prevent normal system (e.g., I/O) functioning.

The present invention provides a secondary mechanism for perform a system function such as I/O functionality. This is accomplished through a set of primitive functions corresponding to a set of system services. The primitive functions, although "equivalent" to the set of system services, are reduced in functionality as well as performance. In this manner, a secondary mechanism for performing the set of system services is made available in the event of a system error.

It is important to note that the primitive functions are made available via requests that are sent automatically (without human intervention). These requests are sent by the set of system services (or stack of system services). In response to the requests, the appropriate primitive function(s) are identified. None of the cited references, separately or in combination, discloses or suggests the claimed invention.

Claims 1, 2, 7-20, 22-24, 26, 29, 35, 38, 43 and 49

Claims 1 and 22, as amended, are directed to a method and computer-readable medium, respectively, for “providing replacement functions for the set of software system services, comprising:

automatically sending a request for a primitive function from one of the set of software system services to another one of the set of software system services, the primitive function replicating the another one of the set of software system services receiving the request for the primitive function in a manner such that implementation of the primitive function reduces or eliminates reliance on one or more system functions capable of becoming non-functional in the event of a system error; and

receiving an identifier associated with the requested primitive function at the one of the set of software system services sending the request for the primitive function from another one of the set of software system services in response to the request, thereby enabling the one of the set of software system services to call the primitive function via the identifier associated with the requested primitive function instead of the another one of the set of software system services.”

Claims 7 and 23, as amended, recite a method or apparatus for providing replacement functions for a stack of software system services, the stack of system services including one or more layers, each layer representing one of the software system services, wherein lower layers provide services to upper layers in the stack, comprising:

automatically sending a primitive function request for a primitive function down from a first one of the layers in the stack of software system services to a second one of the layers in the stack of software system services, the primitive function replicating the system service associated with the second one of the layers in the stack of software system services in a

manner such that implementation of the primitive function reduces or eliminates reliance on one or more system functions capable of becoming non-functional in the event of a system error;

returning primitive function information associated with the primitive function to the first one of the layers sending the primitive function request; and

storing the primitive function information to enable the first one of the layers in the stack of system services sending the primitive function request to communicate with the primitive function associated with the second one of the layers in the stack of software system services.

Similarly, claim 13 recites: “A system for providing replacement system functions in a computer system, comprising:

a set of software components providing a set of services;

a set of primitive software functions associated with the set of services, the set of primitive software functions replicating the set of services, wherein each of the set of primitive software functions eliminates or reduces reliance on one or more system functions that are capable of becoming non-functional in the event of a system error; and

a primitive function request mechanism adapted for being called by one of the set of software components providing the set of services and returning one or more identifiers associated with one or more of the set of primitive software functions to the one of the set of software components calling the primitive function request mechanism, thereby enabling the one of the set of software components calling the primitive function request mechanism to call the one or more of the set of primitive software functions via the returned one or more identifiers.”

Preisler neither discloses nor suggests communication between two system services, as claimed. Col. 8, line 45 – col. 9, line 29 indicate that item 20 is a patch site, item 50 is a patch area, and item 70 includes checking codes. Specifically, col. 8, lines 53-57 indicate that the “run-time checking (RTC) module scans each and every individual instruction that needs to be patched...and the original instructions are then replaced by unconditional branch instructions to the patch area.” Col. 9, lines 1-5 indicate that “there is a custom section of the patch area 50 that is assigned to the whole load object for each patch site 20 and each patch

site 20 is replaced with a branch to its own custom section in the patch area 50.” In other words, a module (RTC module) external to the patch site 20 or patch area 50 performs the “run-time checking.” In other words, a request for a “replacement” is not sent by a patch site 20 to another patch site 20. Moreover, a patch site 20 does not return an identifier of a section in the patch area 50 in response to such a request for a “replacement.” Stated another way, a patch site 20 does not send a request to another patch site for the receiving patch site’s replacement. This is further emphasized at col. 6, lines 22-25 and 27-37, cited by the Examiner, which state “Once the debugger program has received a list of the load objects, it will scan the load objects, searching for instructions that it is going to patch later on. The only part of the load object the debugger program looks at during this instruction-by-instruction scan are the instructions themselves, i.e., the text, but not the data. While the debugger program is identifying the patch sites, the debugger program also accumulates information regarding these patch sites, including patch site address, patch area address, patch type...” Accordingly, Preisler fails to disclose or suggest “sending a request for a primitive function from one of the set of software system services to another one of the set of software system services, the primitive function replicating the another one of the set of software system services in a manner such that implementation of the primitive function reduces or eliminates reliance on one or more system functions capable of becoming non-functional in the event of a system error” or “receiving an identifier associated with the requested primitive function at the one of the set of software system services sending the request for a primitive function from another one of the set of software system services, thereby enabling the one of the set of software system services to call the primitive function via the identifier associated with the requested primitive function instead of the another one of the set of software system services.”

Col. 5, lines 20-25 state that the “in-memory copy of the program 302 changes to a patched program, called “instrumented program” herein. The patches are applied only to this in-memory copy 308 of the program 302 and not to the original program 302 stored on disk 301.” Moreover, col. 5, lines 60-65 state that “the locations that need to be patched are identified as patch sites. Furthermore, the original instructions at these patch sites are then replaced by a branch to patch area instruction.” Patches are known in the art to include a branch or other instruction to a patch area. The instructions in the patch area are merely executed upon execution of the branch instruction. A request is not sent by the original

program for its replacement (e.g., patched program) or a replacement of programs to be called by the original program. In no manner does Preisler disclose or suggest sending a request from the original program for a primitive function replicating the original program (or replicating the subroutines or programs it calls). Moreover, Preisler neither discloses nor suggests sending a request from a section of the original program for a primitive function replicating that section or another section. Thus, Preisler neither discloses nor suggests sending a request from a system service for equivalent lower level services. In fact, as described above, a separate, external RTC module is responsible for “instrumenting” the in-memory copy of the program. As a result, Preisler teaches away from the claimed invention.

The Examiner admits that “Preisler doesn’t explicitly disclose receiving the request for the primitive function services and sending the request for the primitive function. Preisler does disclose meeting requests for executing code and in turn updating the required instructions (7:25 – 35).” The Examiner seeks to cure the deficiencies of Preisler with Edwards, stating that “Edwards discloses in an analogous art sending and receiving requests regarding needed functions in a similar environment (Col. 5 and 6, lines 20-30). Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine Preisler and Edwards because, being able to send and receive requests would enable updating or replacing code as needed.” Applicant respectfully traverses this assertion.

Edwards does disclose the use of request/reply messages. However, as set forth above with respect to Preisler, the request messages are not sent or received by software modules being replaced. Similarly, an identifier of a replacement function is not provided in response to a request message. In fact, the request/reply messages of Edwards merely provide debugging functionality (e.g., event tracing and counting). See col. 5, lines 53-67. In other words, debuggers merely enable a user to debug non-functional software. Edwards does not disclose request/reply messages that are automatically sent or received by the software that is being replaced by primitive functions. Moreover, col. 6, lines 21-30 indicate that the request messages of Edwards are only sent when a user sends a debug request, not automatically. As such, Edwards teaches away from the claimed invention.

It is important to note that both Preisler and Edwards relate to debuggers and debugging applications. Thus, Preisler and Edwards enable the debugging of software. In contrast, the claimed invention, in accordance with various embodiments, relates to replacing

system functions that may support debugging. For instance, system services supporting input/output functionality may be preserved, enabling debugging to be performed. In other words, the claimed invention does not relate to replacing instructions during debugging, but replacing system functionality (e.g., I/O functionality) that supports debugging. Applicant respectfully submits that neither of the cited references, separately or in combination, discloses or suggests the problem created by inoperable system functionality necessary for operation of a debugger or a solution to this problem. In fact, both Preisler and Edwards assume that the underlying system functionality is operational. Accordingly, both Preisler and Edwards teach away from the claimed invention.

It is also important to note that the combination of the cited references would fail to operate as claimed. In fact, the combination of the cited references would merely result in a debugger that performs functions such as debugging or tracing “target” applications. Combination of the cited references would fail to result in an automated system enabling system functions such as I/O functions to access their own replacement functions. Accordingly, Applicant respectfully submits that the claims are patentable over the cited references.

The dependent claims are dependent upon one of the independent claims, and are therefore patentable for at least the reasons set forth above. The additional limitations in the independent or dependent claims are not further discussed as the above limitations are sufficient to distinguish the claimed invention from the cited reference. Accordingly, Applicants contend that the rejection is unsupported by the art and should be withdrawn. Thus, it is respectfully requested that the Examiner withdraw the rejection of the dependent claims under 35 USC §103.

Claims 3-6, 21, 25, 27, 28, 33, 34, 36, 37, 42, 44, 45, 47 and 48

Similarly, claim 4, as amended, is directed to a method of “providing replacement functions for a stack of software system services, the stack of software system services including one or more layers, each layer representing one of the software system services, wherein lower layers provide services to upper layers in the stack, the method comprising:

automatically sending a primitive function request for a primitive function down from one of the layers of the stack of software system services to another one of the layers in the stack of software system services, the primitive function replicating the system service

associated with the another one of the layers in the stack receiving the primitive function request;

when the another one of the layers receiving the primitive function request is responsible for performing at least one of input and output, returning a primitive function identifier associated with the primitive function to the one of the layers of the stack of software system services sending the primitive function request in response to the primitive function request.”

As set forth above, Applicant respectfully submits that claim 4 is patentable over Preisler in view of Edwards. It appears that the Examiner has cited Ahlin merely because it discloses input and output functionality. However, as set forth in Ahlin, the input/output functionality may be implemented in “BIOS” software, which “can be downloaded from the network host when needed, a process which might take place on the order of several times per year.” See col. 11, line 65-col. 12, line 14. In other words, the BIOS software must be downloaded by a user. Thus, the software does not automatically access its own replacement functions upon which it depends. Thus, Ahlin teaches away from the claimed invention. Moreover, Ahlin fails to cure the deficiencies of Preisler and Edwards.

None of the cited references, separately or in combination, discloses or suggests a set of primitive software functions associated with a set of services, the set of primitive software functions replicating the set of services, wherein each of the set of primitive software functions eliminates or reduces reliance on one or more system functions that are capable of becoming non-functional in the event of a system error. The set of primitive functions are particularly useful, for example, when testing system services providing input and/or output functionality or keyboard functionality, as recited in claims 45 and 47.

Claims 18, 29, 38 and 49; claims 30, 32, 39, 41, 50 and 52

Primitive functions may be implemented, for example, by implementing polling rather than interrupts, or by implementing delay loops rather than timers, as recited in claims 49-52, which are less susceptible to problems (and more likely to be functional) in the event of a system error. In no manner do the cited references, separately or in combination, disclose or suggest the calling of a primitive function request mechanism by a software component providing a service. Moreover, none of the cited references discloses or suggests

providing an identifier associated with a primitive function to the software component calling the primitive function request mechanism, thereby enabling the software component calling the primitive function request mechanism to call the primitive software function via the returned one or more identifiers. Thus, Kirouac and Halpern fail to cure the deficiencies of the primary references.

Claims 31, 40 and 51

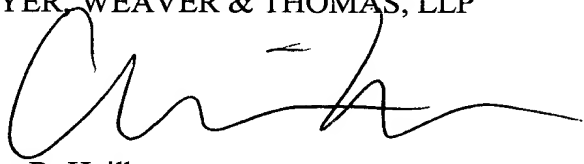
Glasser fails to cure the deficiencies of the primary references. Specifically, although discloses the use of delay loops, neither of the cited references discloses the use of these functions in the manner claimed. More specifically, the claimed invention enables primitive functions replicating system services to replace system services in the event of a system error. These system functions and associated primitive functions may implement, for example, input and/or output functionality using mechanisms such as delay loops. In this manner, such functionality is retained, even in the event of a system error, thereby enabling debugging through input/output functionality to be accomplished. None of the cited references, separately or in combination, discloses or suggests the problem of debugging system services after a system error has occurred. Moreover, none of the cited references, separately or in combination, discloses or suggests the inoperability of input/output or keyboard functionality or potential solutions in the event of a system (e.g., hardware) error. Accordingly, Applicant respectfully submits that the pending claims are patentable over the cited art.

If there are any issues remaining which the Examiner believes could be resolved through either a Supplemental Response or an Examiner's Amendment, the Examiner is respectfully requested to contact the undersigned attorney at the telephone number listed below.

Applicants hereby petition for any extension of time which may be required to maintain the pendency of this case, and any required fee for such extension or any further fee

required in connection with the filing of this Amendment is to be charged to Deposit Account No. 50-0388 (Order No. SUN1P376).

Respectfully submitted,
BEYER, WEAVER & THOMAS, LLP

A handwritten signature in black ink, appearing to read 'Elise R. Heilbrunn', written over the firm name.

Elise R. Heilbrunn
Reg. No. 42,649

BEYER, WEAVER & THOMAS, LLP
P.O. Box 70250
Oakland, CA 94612-0250
Tel: (510) 663-1100
Fax: (510) 663-0920